Wireshark

- GUI network protocol analyzer and packet sniffer
- Uses libpcap standard library for opening and capturing network traffic
- Customizable dissectors (modules) for proprietary protocols
- Multi-platform support including Linux, Mac, Windows, etc.

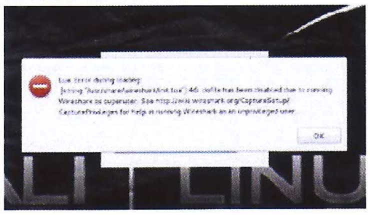
Wireshark is available for most Unix platforms and later Windows platforms (XP, Vista, 7, 2003, 2008).

Wireshark is THE standard for performing network protocol analysis

Some examples as to how Wireshark is used are:

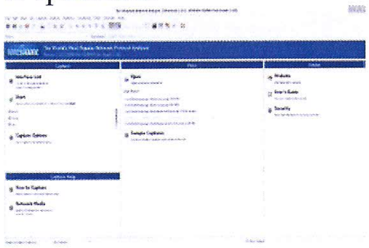
- Network administrators use it to troubleshoot network issues
- Network security engineers use it to examine security problems
- Developers use it to debug protocol implementations
- People use it to learn network protocol internals.

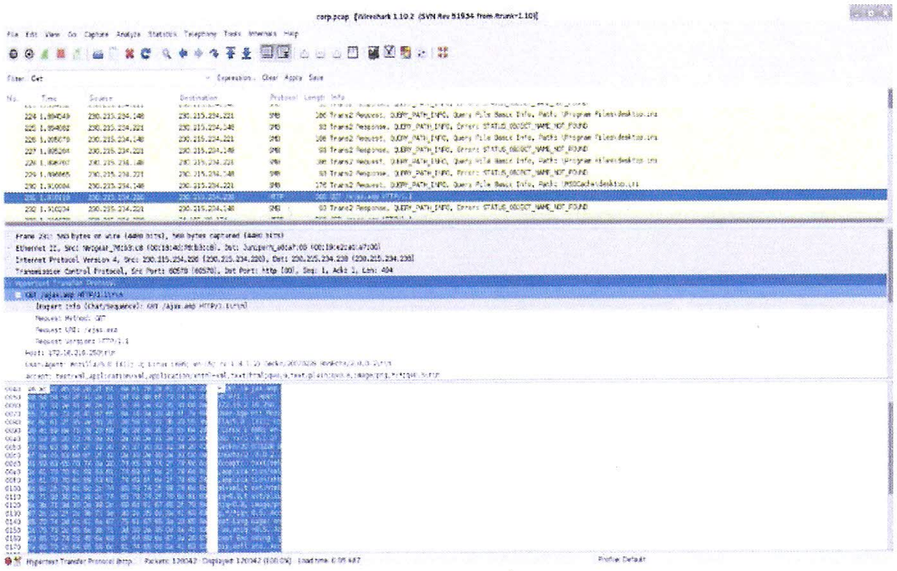
Warning when Wireshark is run as root/admin.



Lua: Error during loading:
[string "/usr/share/wireshark/init.lua"]:46: dofile has been disabled due to running Wireshark as superuser. See http://wiki.wireshark.org/CaptureSetup/CapturePrivileges for help in running Wireshark as an unprivileged user.

Welcome screen if no file is specified.





corp.pcap [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter: Get
Expression... Clear Apply Save
No. Time Source Destination Protocol Length Info

Frame 231: 548 bytes on wire (4384 bits), 548 bytes captured (4384 bits)
Ethernet II, Src: Netgear_... Dst: Juniper ...
Internet Protocol Version 4, Src: 230.215.234.... Dst: 230.215.234.230 (230.215.234.230)
Transmission Control Protocol, Src Port: 60570 (60570), Dst Port: http (80), Seq: 1, Ack: 1, Len: 494
Hypertext Transfer Protocol
GET /ajax.asp HTTP/1.1\r\n
Request Method: GET
Request URI: /ajax.asp
Request Version: HTTP/1.1

Wireshark consists of parts that are common to many GUI programs:

- *Menu* - used to start actions
- *Main toolbar* - provides quick access
- *Filter toolbar* - provides a way to directly manipulate the current display filter
- *Packet list pane* - displays a summary of each packet captured
- *Packet details pane* - displays the packet selected, more detail
- *Packets bytes pane* - displays the data from packet selected
- *Status bar* - shows some detailed information about current program state & data.

Security Notes:

- Vulnerabilities in Wireshark could leave your system at risk of compromise if used on active networks
- Not required to run with root privilege
- Long-term traffic monitoring should be done with "tcpdump."

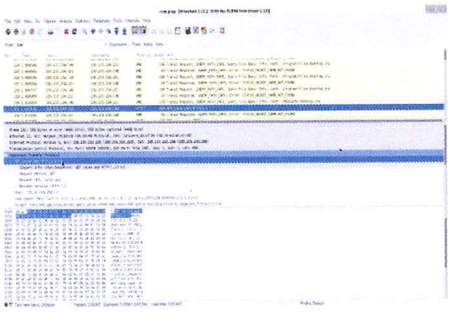
Rule of Thumb:

- Capture with tcpdump and analyze with Wireshark **using a normal user account**. Wireshark is a large, complicated utility and has been known to have vulnerabilities caused by specially crafted malformed packets. Running Wireshark as root (aka admin) puts you at a higher risk level.

See the following for more information:

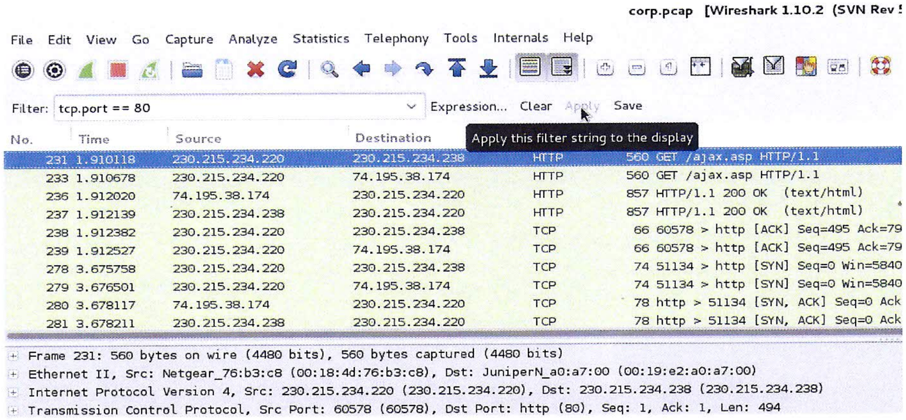
http://wiki.wireshark.org/Security

Selecting text in any of the three main information panes will highlight the associated text and/or packet in the other panes.



Wireshark display filter examples

- tcp.port == 80
- ftp
- ip.addr == 192.168.10.97
- ip.addr == 192.168.10.97 && tcp.port == 80


corp.pcap [Wireshark 1.10.2 (SVN Rev !
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save
Apply this filter string to the display

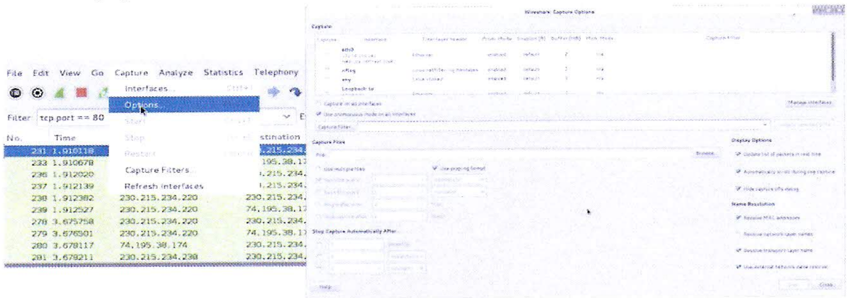
Filter: tcp.port == 80

```
No.     Time       Source             Destination
  231 1.910118   230.215.234.220    230.215.234.238    HTTP    560 GET /ajax.asp HTTP/1.1
  233 1.910678   230.215.234.220    74.195.38.174      HTTP    560 GET /ajax.asp HTTP/1.1
  236 1.912020   74.195.38.174      230.215.234.220    HTTP    857 HTTP/1.1 200 OK  (text/html)
  237 1.912139   230.215.234.238    230.215.234.220    HTTP    857 HTTP/1.1 200 OK  (text/html)
  238 1.912382   230.215.234.220    230.215.234.238    TCP      66 60578 > http [ACK] Seq=495 Ack=79
  239 1.912527   230.215.234.220    74.195.38.174      TCP      66 60578 > http [ACK] Seq=495 Ack=79
  278 3.675758   230.215.234.220    230.215.234.238    TCP      74 51134 > http [SYN] Seq=0 Win=5840
  279 3.676501   230.215.234.220    74.195.38.174      TCP      74 51134 > http [SYN] Seq=0 Win=5840
  280 3.678117   74.195.38.174      230.215.234.220    TCP      78 http > 51134 [SYN, ACK] Seq=0 Ack
  281 3.678211   230.215.234.238    230.215.234.220    TCP      78 http > 51134 [SYN, ACK] Seq=0 Ack

+ Frame 231: 560 bytes on wire (4480 bits), 560 bytes captured (4480 bits)
+ Ethernet II, Src: Netgear_76:b3:c8 (00:18:4d:76:b3:c8), Dst: JuniperN_a0:a7:00 (00:19:e2:a0:a7:00)
+ Internet Protocol Version 4, Src: 230.215.234.220 (230.215.234.220), Dst: 230.215.234.238 (230.215.234.238)
+ Transmission Control Protocol, Src Port: 60578 (60578), Dst Port: http (80), Seq: 1, Ack: 1, Len: 494
```

Reference for BPF syntax: http://biot.com/capstats/bpf.html

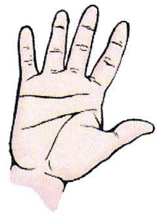
Wireshark capture filter examples

- Uses Berkeley Packet Filter (BPF) syntax
- host 10.10.0.20
- tcp port 80



```
231 1.910118                      .215.234.
233 1.910678                      195.38.17
236 1.912020                      .215.234.
237 1.912139                      .215.234.
238 1.912382  230.215.234.220    230.215.234.
239 1.912527  230.215.234.220    74.195.38.17
278 3.675758  230.215.234.220    230.215.234.
279 3.676501  230.215.234.220    74.195.38.17
280 3.678117  74.195.38.174      230.215.234.
281 3.678211  230.215.234.238    230.215.234.
```

Tcpdump and Wireshark Hands-on Exercise



Tcpdump

Tcpdump is a simple tool that allows you to capture all traffic on the network interface whether it is destined for your computer or not. However, because of the switched network environment you will only have traffic generated from or destined for your computer and broadcast traffic on your network interface. As a result, we need to generate some traffic to be captured by tcpdump. This can be done by using the nmap command that will be discussed in detail later. By simultaneously running the nmap and tcpdump commands, a network traffic capture can be generated.

Open a Terminal window and ensure you are in the Desktop directory. If your prompt does not show root@kali:~/Desktop#, then type *cd /root/Desktop* to change into the proper directory.

To start the network traffic capture, type the following tcpdump command at the prompt:

```
tcpdump –s 0 –i eth0 –w 301exercise.pcap
```

While tcpdump is running, run the nmap command to generate some interesting traffic. This can be done by typing the following nmap command in a **new** command shell:

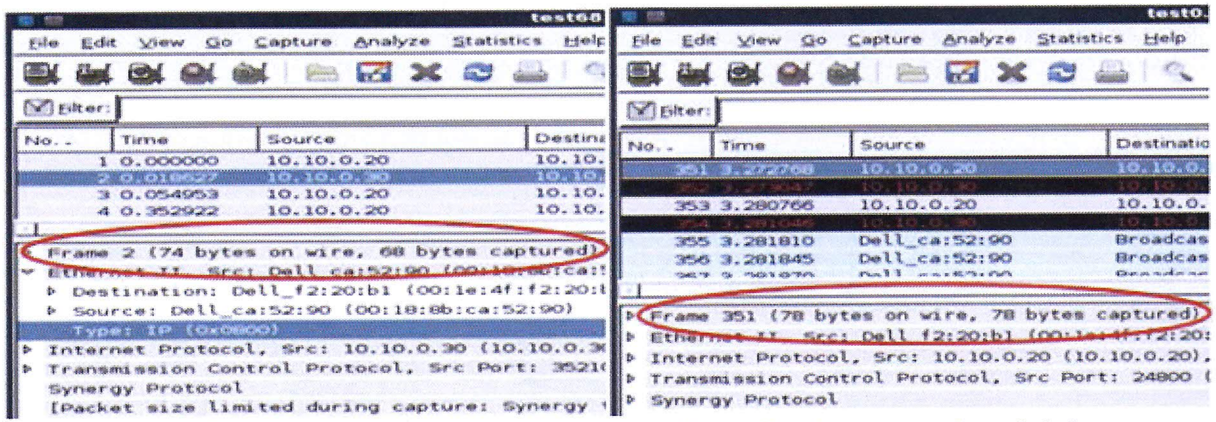
```
nmap –n <your network>.1-99
```

Once the nmap command completes, stop the tcpdump capture by pressing ctrl+c. This should have created 301exercise.pcap on the desktop. This file will be analyzed later using Wireshark.

A snap length of –s 0 means that tcpdump will capture the entire contents of each packet; a snap length of –s 68 will only capture the first 68 bytes of each packet. Try creating another traffic capture with a different snap length by modifying the above steps. Be sure to provide a different file name so that you can compare the two traffic captures later.

Wireshark

Use Wireshark to open and analyze the network traffic captures created during the tcpdump exercises. This can be done by **double** clicking each of the files on the Desktop; this will automatically launch Wireshark with the selected pcap file. You should notice that the number of bytes captured for each packet is different between the two pcap files. Figure 7 illustrates the difference in snap lengths.


test68
File Edit View Go Capture Analyze Statistics Help
Filter:
test0.
File Edit View Go Capture Analyze Statistics Help
Filter:

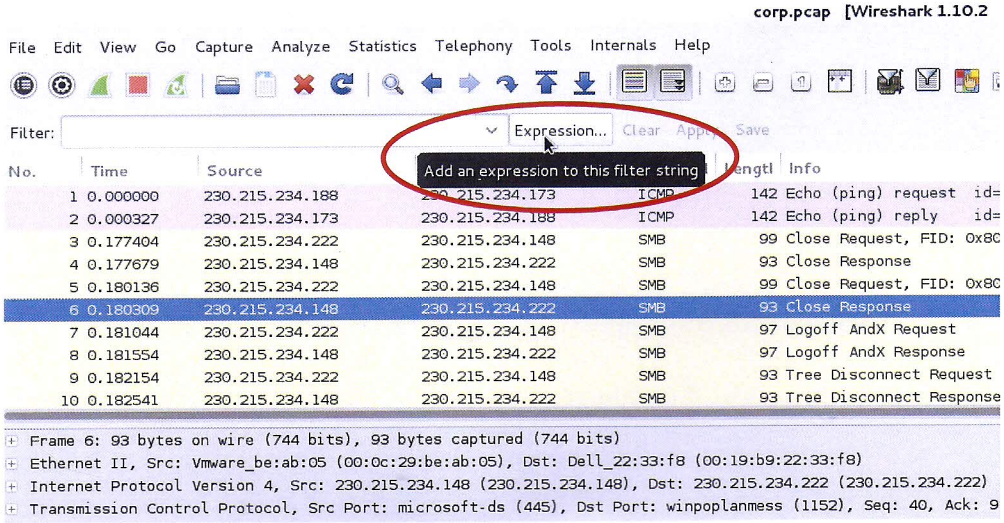
```
No. .  Time        Source       Destin
     1 0.000000    10.10.0.20   10.10.
     2 0.018627    10.10.0.30   10.10.
     3 0.054953    10.10.0.20   10.10.
     4 0.352922    10.10.0.20   10.10.

Frame 2 (74 bytes on wire, 68 bytes captured)
Ethernet II, Src: Dell_ca:52:90 (00:18:8b:ca:5
  Destination: Dell_f2:20:b1 (00:1e:4f:f2:20:
  Source: Dell_ca:52:90 (00:18:8b:ca:52:90)
  Type: IP (0x0800)
Internet Protocol, Src: 10.10.0.30 (10.10.0.3
Transmission Control Protocol, Src Port: 3521
Synergy Protocol
[Packet size limited during capture: Synergy
```

```
No. .  Time        Source          Destinatio
   351 3.277768    10.10.0.20      10.10.0.
   352 3.279047    10.10.0.30      10.10.0.
   353 3.280766    10.10.0.20      10.10.0.
   354 3.281046    10.10.0.30      10.10.0.
   355 3.281810    Dell_ca:52:90   Broadcas
   356 3.281845    Dell_ca:52:90   Broadcas

Frame 351 (78 bytes on wire, 78 bytes captured)
Ethernet II, Src: Dell_f2:20:b1 (00:1e:4f:f2:20:
Internet Protocol, Src: 10.10.0.20 (10.10.0.20),
Transmission Control Protocol, Src Port: 24800 (
Synergy Protocol
```

Figure 7 – Left: Tcpdump with snap length 68 (bytes). Right Tcpdump snap length 0 (i.e., capture entire packet).

On the Desktop, there is a folder named **pcap_files**. Within this folder there are pcap files that contain anonymized network traffic captures of SCADA and corporate environments. There are also captures taken during the Metasploit exercises you'll be doing in a later session. Find ICS protocols, identify corporate business services, extract information from clear text protocols, and anything else interesting or unusual. To help in the analysis of the traffic capture, generate display filters to show you the packets of interest. This is done by entering the display filter syntax directly into the filter box if you know the syntax, or by using the GUI-based expression builder that will generate the appropriate display filter syntax for you.

Next, is an example of how to create a display filter for TCP port 80 using the expression builder. First, click the Expression icon that is next to the Filter box as shown in Figure 8.


corp.pcap [Wireshark 1.10.2
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
Clear
Apply
Save
Add an expression to this filter string

```
No.  Time       Source           Destination      Protocol  Length Info
   1 0.000000   230.215.234.188  230.215.234.173  ICMP      142 Echo (ping) request  id=
   2 0.000327   230.215.234.173  230.215.234.188  ICMP      142 Echo (ping) reply    id=
   3 0.177404   230.215.234.222  230.215.234.148  SMB        99 Close Request, FID: 0x8C
   4 0.177679   230.215.234.148  230.215.234.222  SMB        93 Close Response
   5 0.180136   230.215.234.222  230.215.234.148  SMB        99 Close Request, FID: 0x8C
   6 0.180309   230.215.234.148  230.215.234.222  SMB        93 Close Response
   7 0.181044   230.215.234.222  230.215.234.148  SMB        97 Logoff AndX Request
   8 0.181554   230.215.234.148  230.215.234.222  SMB        97 Logoff AndX Response
   9 0.182154   230.215.234.222  230.215.234.148  SMB        93 Tree Disconnect Request
  10 0.182541   230.215.234.148  230.215.234.222  SMB        93 Tree Disconnect Response

+ Frame 6: 93 bytes on wire (744 bits), 93 bytes captured (744 bits)
+ Ethernet II, Src: Vmware_be:ab:05 (00:0c:29:be:ab:05), Dst: Dell_22:33:f8 (00:19:b9:22:33:f8)
+ Internet Protocol Version 4, Src: 230.215.234.148 (230.215.234.148), Dst: 230.215.234.222 (230.215.234.222)
+ Transmission Control Protocol, Src Port: microsoft-ds (445), Dst Port: winpoplanmess (1152), Seq: 40, Ack: 9
```

Figure 8 – Circled in red are the Expression builder and Apply buttons for display filters.

Once the expression builder window appears, shown in Figure 9, scroll down to the **TCP-Transmission Control Protocol** drop down menu. (Note: this will almost be at the bottom of the selection window.) Open the TCP drop down menu and select "tcp.port" then select "==" from the Relation column, and finally type "80" in the Value box at the upper right.

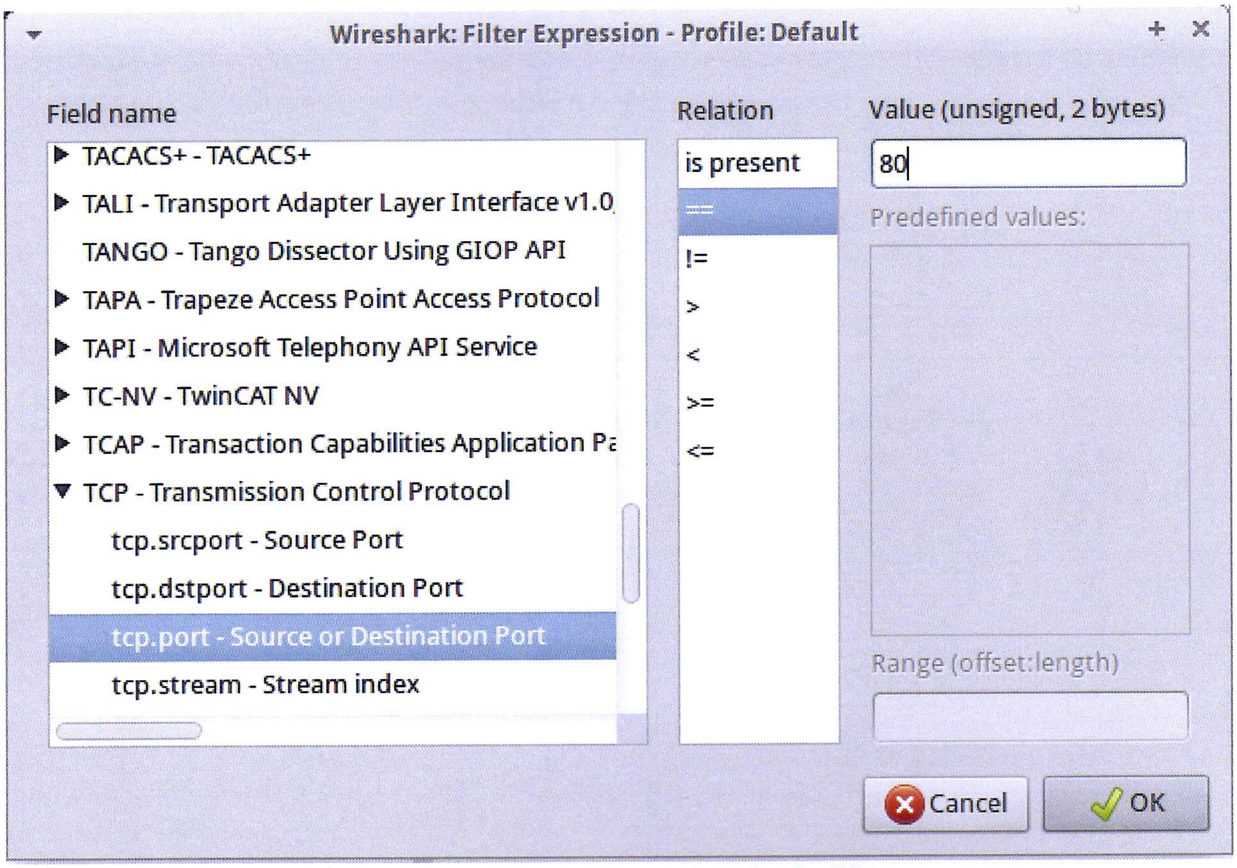

Wireshark: Filter Expression - Profile: Default
Field name
▶ TACACS+ - TACACS+
▶ TALI - Transport Adapter Layer Interface v1.0
TANGO - Tango Dissector Using GIOP API
▶ TAPA - Trapeze Access Point Access Protocol
▶ TAPI - Microsoft Telephony API Service
▶ TC-NV - TwinCAT NV
▶ TCAP - Transaction Capabilities Application Pa
▼ TCP - Transmission Control Protocol
tcp.srcport - Source Port
tcp.dstport - Destination Port
tcp.port - Source or Destination Port
tcp.stream - Stream index
Relation
is present
==
!=
>
<
>=
<=
Value (unsigned, 2 bytes)
80
Predefined values:
Range (offset:length)
Cancel
OK

Figure 9 – Display filter expression builder

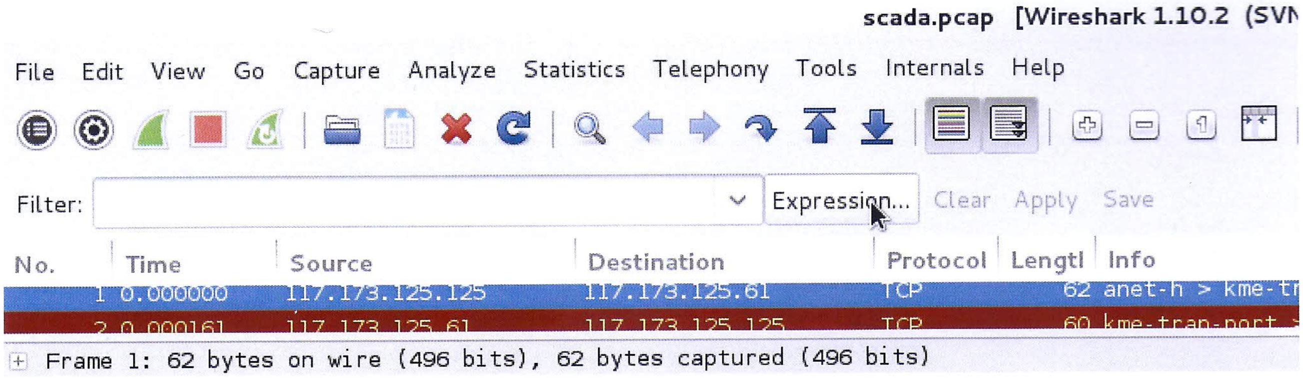
Finish by clicking OK, and the display filter will automatically be generated and entered in the Filter box. To activate the filter, press the "Apply" icon next to the "Expression" icon. Once the filter is applied, the only packets shown will either have a source or destination port of 80.

Continue by creating your own display filter combinations. You can use the relation operators && and || to combine filters. For example tcp.port == 80 && ip.addr == 1.2.3.21 is a combination of two filters.

If you want to follow a TCP stream for a given packet, right-click the packet and select "Follow TCP Stream." This will generate a display filter that will show you only that TCP stream. Figure 10 on the next page is a reference for some of the basic Wireshark display filter syntax. Use this table or the expression builder (Figure 9) to help you create your own display filter combinations.

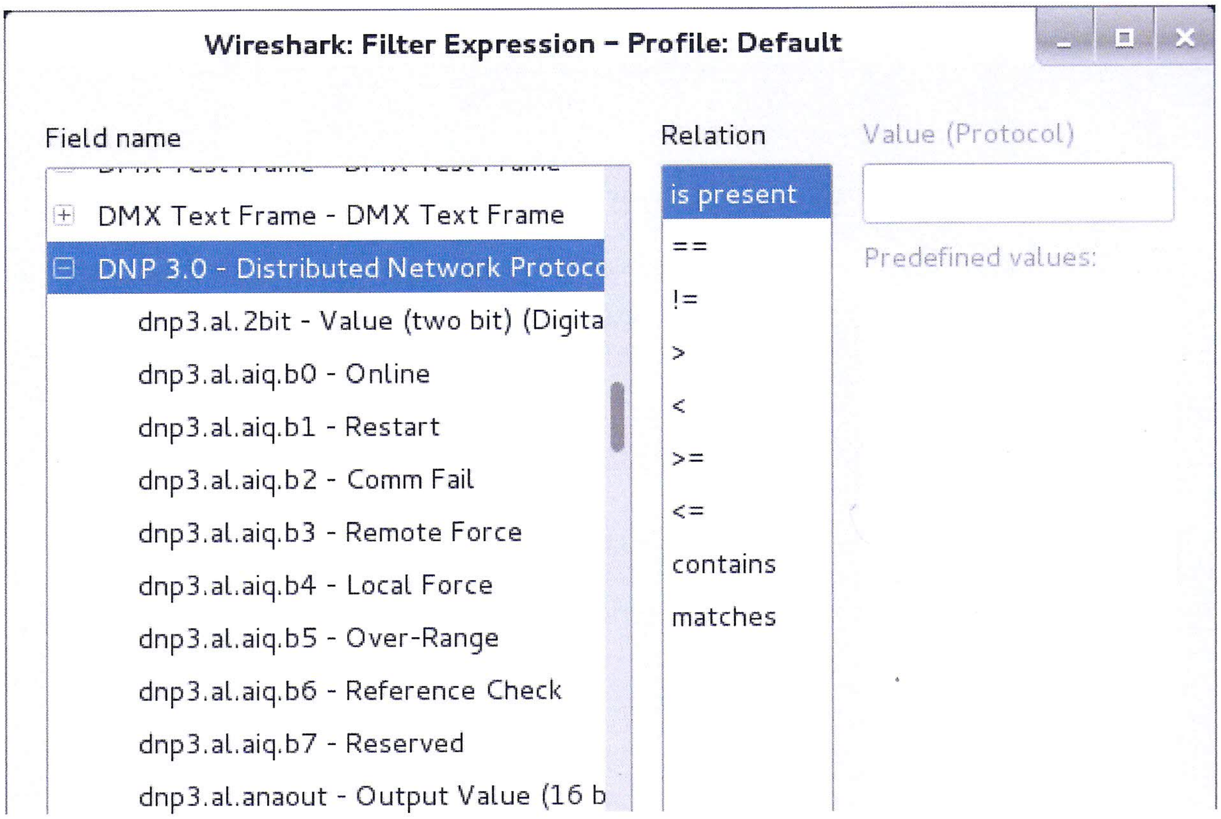
Displaying packets with DNP3 properties.

Wireshark has built in display filters for many popular network and application protocols. One group of display filters is for DNP3. To access these filters, click on the *Expression* button next to the filter window.


scada.pcap [Wireshark 1.10.2 (SVN
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression...
Clear
Apply
Save

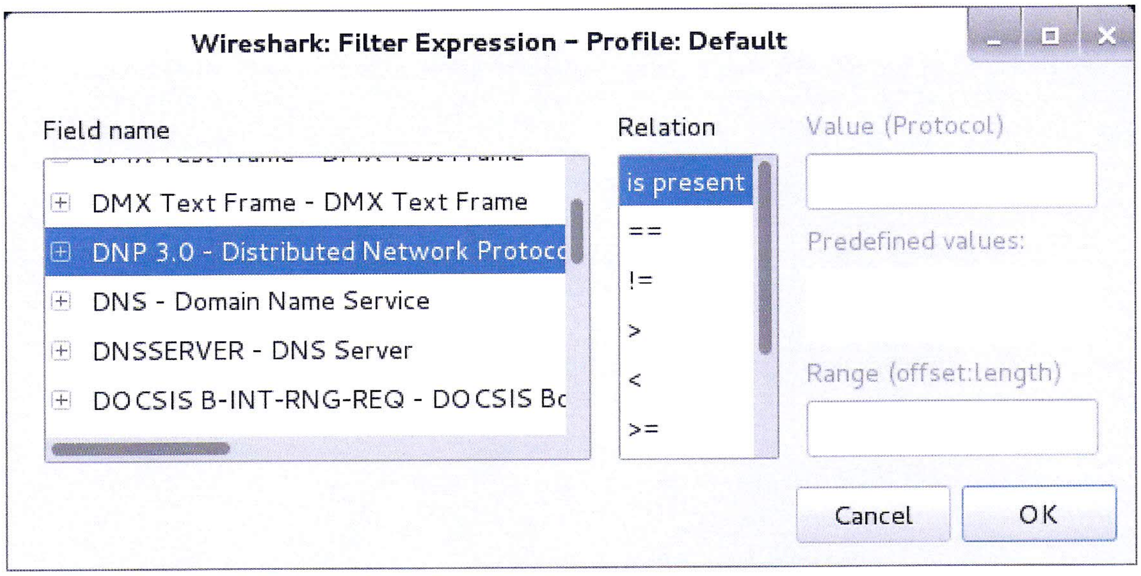
```
No.   Time        Source            Destination       Protocol Lengtl Info
    1 0.000000    117.173.125.125   117.173.125.61    TCP         62 anet-h > kme-tr
    2 0.000161    117.173.125.61    117.173.125.125   TCP         60 kme-trap-port >
+ Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)
```

A new pop-up search window will appear. Search down this list for DNP 3.0


Wireshark: Filter Expression – Profile: Default
Field name
DMX Text Frame - DMX Text Frame
DNP 3.0 - Distributed Network Protoco
dnp3.al.2bit - Value (two bit) (Digita
dnp3.al.aiq.b0 - Online
dnp3.al.aiq.b1 - Restart
dnp3.al.aiq.b2 - Comm Fail
dnp3.al.aiq.b3 - Remote Force
dnp3.al.aiq.b4 - Local Force
dnp3.al.aiq.b5 - Over-Range
dnp3.al.aiq.b6 - Reference Check
dnp3.al.aiq.b7 - Reserved
dnp3.al.anaout - Output Value (16 b
Relation
is present
==
!=
>
<
>=
<=
contains
matches
Value (Protocol)
Predefined values:

Homeland Security

When you click on the + next to the entry, a list of DNP3 protocol fields will appear.



Wireshark: Filter Expression - Profile: Default

Field name

- DMX Text Frame - DMX Text Frame
- DNP 3.0 - Distributed Network Protoco
- DNS - Domain Name Service
- DNSSERVER - DNS Server
- DOCSIS B-INT-RNG-REQ - DOCSIS Bc

Relation

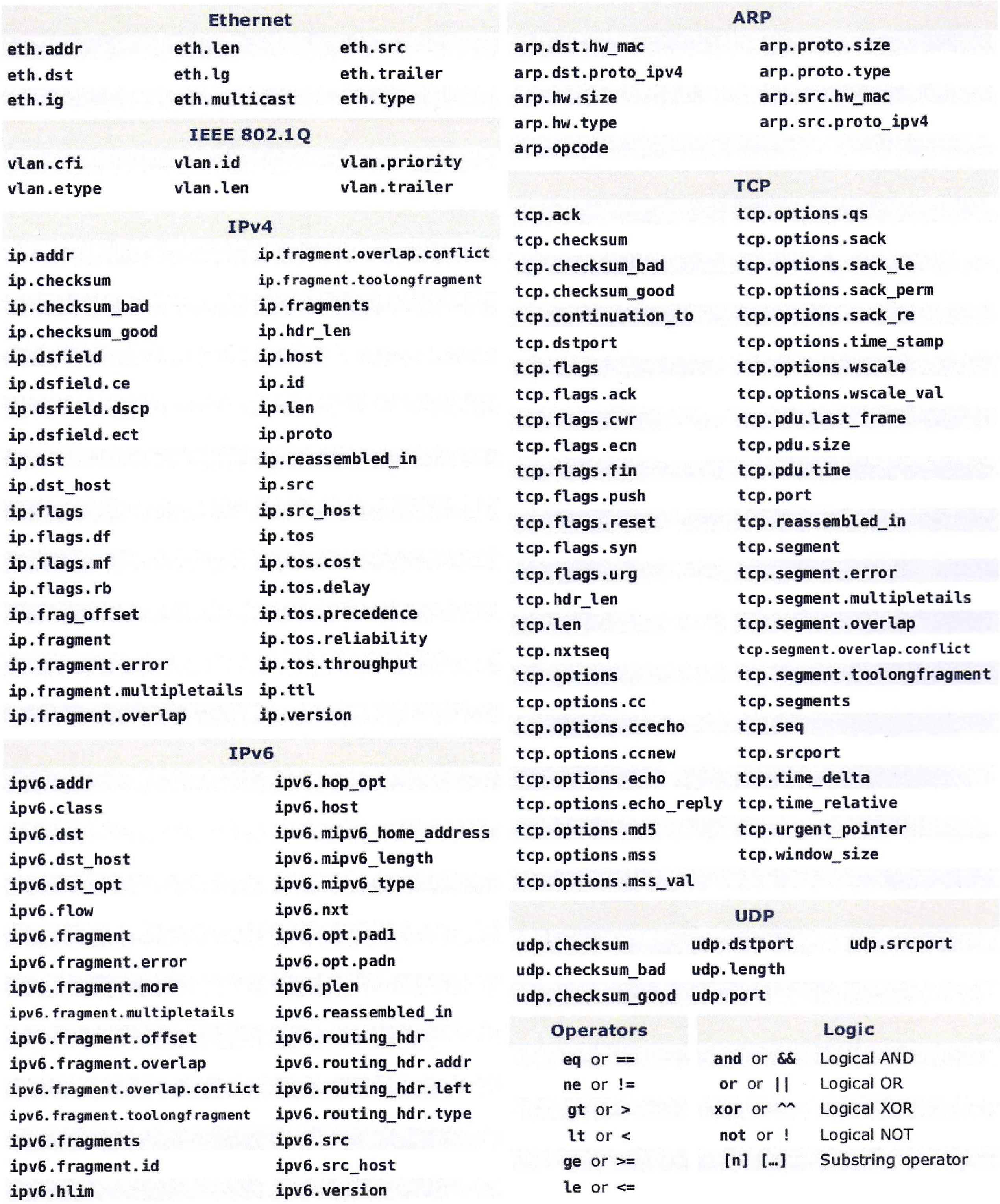
- is present
- ==
- !=
- >
- <
- >=

Value (Protocol)

Predefined values:

Range (offset:length)

You can then select a parameter you want to filter on, the relation operator, and optionally a value if required. Once you are satisfied with your settings, click **OK** at the bottom. The window will disappear and the display filter will bet set in the expression window. Now click on **Apply Button. Wireshark** will then research the packets to match your filter expression.



Ethernet

- eth.addr
- eth.dst
- eth.ig
- eth.len
- eth.lg
- eth.multicast
- eth.src
- eth.trailer
- eth.type

IEEE 802.1Q

- vlan.cfi
- vlan.etype
- vlan.id
- vlan.len
- vlan.priority
- vlan.trailer

IPv4

- ip.addr
- ip.checksum
- ip.checksum_bad
- ip.checksum_good
- ip.dsfield
- ip.dsfield.ce
- ip.dsfield.dscp
- ip.dsfield.ect
- ip.dst
- ip.dst_host
- ip.flags
- ip.flags.df
- ip.flags.mf
- ip.flags.rb
- ip.frag_offset
- ip.fragment
- ip.fragment.error
- ip.fragment.multipletails
- ip.fragment.overlap
- ip.fragment.overlap.conflict
- ip.fragment.toolongfragment
- ip.fragments
- ip.hdr_len
- ip.host
- ip.id
- ip.len
- ip.proto
- ip.reassembled_in
- ip.src
- ip.src_host
- ip.tos
- ip.tos.cost
- ip.tos.delay
- ip.tos.precedence
- ip.tos.reliability
- ip.tos.throughput
- ip.ttl
- ip.version

IPv6

- ipv6.addr
- ipv6.class
- ipv6.dst
- ipv6.dst_host
- ipv6.dst_opt
- ipv6.flow
- ipv6.fragment
- ipv6.fragment.error
- ipv6.fragment.more
- ipv6.fragment.multipletails
- ipv6.fragment.offset
- ipv6.fragment.overlap
- ipv6.fragment.overlap.conflict
- ipv6.fragment.toolongfragment
- ipv6.fragments
- ipv6.fragment.id
- ipv6.hlim
- ipv6.hop_opt
- ipv6.host
- ipv6.mipv6_home_address
- ipv6.mipv6_length
- ipv6.mipv6_type
- ipv6.nxt
- ipv6.opt.pad1
- ipv6.opt.padn
- ipv6.plen
- ipv6.reassembled_in
- ipv6.routing_hdr
- ipv6.routing_hdr.addr
- ipv6.routing_hdr.left
- ipv6.routing_hdr.type
- ipv6.src
- ipv6.src_host
- ipv6.version

ARP

- arp.dst.hw_mac
- arp.dst.proto_ipv4
- arp.hw.size
- arp.hw.type
- arp.opcode
- arp.proto.size
- arp.proto.type
- arp.src.hw_mac
- arp.src.proto_ipv4

TCP

- tcp.ack
- tcp.checksum
- tcp.checksum_bad
- tcp.checksum_good
- tcp.continuation_to
- tcp.dstport
- tcp.flags
- tcp.flags.ack
- tcp.flags.cwr
- tcp.flags.ecn
- tcp.flags.fin
- tcp.flags.push
- tcp.flags.reset
- tcp.flags.syn
- tcp.flags.urg
- tcp.hdr_len
- tcp.len
- tcp.nxtseq
- tcp.options
- tcp.options.cc
- tcp.options.ccecho
- tcp.options.ccnew
- tcp.options.echo
- tcp.options.echo_reply
- tcp.options.md5
- tcp.options.mss
- tcp.options.mss_val
- tcp.options.qs
- tcp.options.sack
- tcp.options.sack_le
- tcp.options.sack_perm
- tcp.options.sack_re
- tcp.options.time_stamp
- tcp.options.wscale
- tcp.options.wscale_val
- tcp.pdu.last_frame
- tcp.pdu.size
- tcp.pdu.time
- tcp.port
- tcp.reassembled_in
- tcp.segment
- tcp.segment.error
- tcp.segment.multipletails
- tcp.segment.overlap
- tcp.segment.overlap.conflict
- tcp.segment.toolongfragment
- tcp.segments
- tcp.seq
- tcp.srcport
- tcp.time_delta
- tcp.time_relative
- tcp.urgent_pointer
- tcp.window_size

UDP

- udp.checksum
- udp.checksum_bad
- udp.checksum_good
- udp.dstport
- udp.length
- udp.port
- udp.srcport

Operators

Operators
eq or ==
ne or !=
gt or >
lt or <
ge or >=
le or <=

Logic

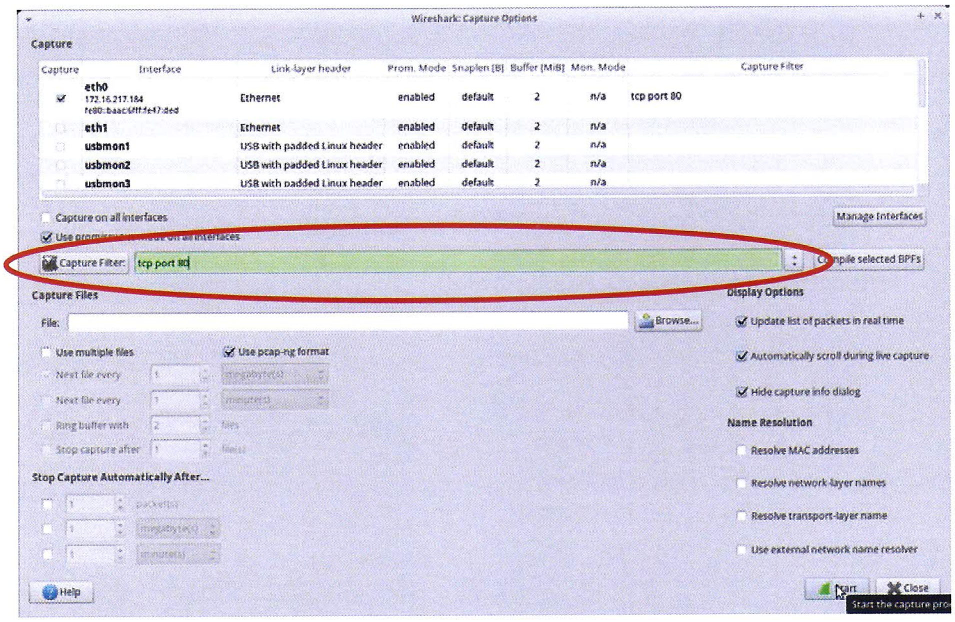
Logic	
and or &&	Logical AND
or or ||	Logical OR
xor or ^^	Logical XOR
not or !	Logical NOT
[n] […]	Substring operator

Figure 10 – Ref: http://packetlife.net/library/cheat-sheets/

The last exercise for Wireshark is to create a network traffic capture (pcap) using Berkley Packet Filters (BPF) to filter the traffic.

Select **Capture** > **Options** on the wireshark menu bar. This will bring up a window similar to Figure 11. At the top of the window, wireshark has listed what it thinks are the usable network interfaces.   Select ***eth0*** from the list.

If the options form renders too big for your screen, you can accomplish this exercise by selecting **Capture > Capture Filters** to input the same information.



Wireshark: Capture Options

Capture

Capture	Interface	Link-layer header	Prom. Mode	Snaplen [B]	Buffer [MiB]	Mon. Mode	Capture Filter
☑	eth0 172.16.217.184 fe80::baac:6fff:fe47:ded	Ethernet	enabled	default	2	n/a	tcp port 80
□	eth1	Ethernet	enabled	default	2	n/a	
□	usbmon1	USB with padded Linux header	enabled	default	2	n/a	
□	usbmon2	USB with padded Linux header	enabled	default	2	n/a	
□	usbmon3	USB with padded Linux header	enabled	default	2	n/a	

□ Capture on all interfaces

☑ Use promiscuous mode on all interfaces

Capture Filter: tcp port 80

Capture Files

File:

□ Use multiple files ☑ Use pcap-ng format

Display Options

☑ Update list of packets in real time

☑ Automatically scroll during live capture

☑ Hide capture info dialog

Name Resolution

□ Resolve MAC addresses

□ Resolve network-layer names

□ Resolve transport-layer name

□ Use external network name resolver

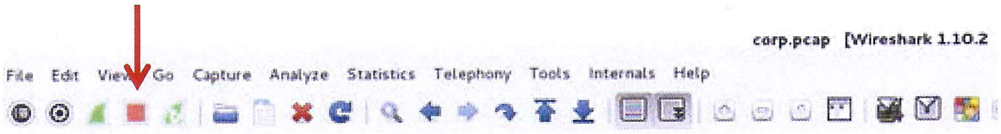
Figure 11 – Capture options.

Capture filters can be specified within a text file and referenced by the filename, or the capture filters can be typed directly into the "Capture Filter" box.

To create a simple capture filter for http traffic, type tcp port 80 into the "Capture Filter" box as shown in Figure 11. This will only capture tcp traffic with a source or destination port of 80. Click the "Start" button at the bottom of the window to start capturing packets. As with tcpdump, we need to run the following nmap command in a command shell to generate some interesting traffic.

```
nmap  –n  <your network>.1-99
```

Once the nmap command completes you can stop Wireshark from capturing packets by clicking the icon with the red box at the top of the Wireshark window.



corp.pcap [Wireshark 1.10.2
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Post-exercise analysis

- What network protocols did you find?

- What ICS-specific protocols did you find?

- Were there plain text protocols?

Passive Discovery Review

- Automated tools, protocol caches, configuration files, history files, etc. are valuable sources for passive network reconnaissance
- Tcpdump and Wireshark are the de facto standards for network sniffing and protocol analysis.

Learning Objective 6

LO6: Discuss active discovery

Again, there are two types on network discovery: passive and active. Now, let's look at active discovery.

Active Discovery

What is ***active*** network discovery?

- Send network packets and wait for a response in order to identify host and network targets
- Can be extremely noisy and easily detected.

Why use ***active*** discovery methods?

- Identify targets that cannot be otherwise identified using passive discovery techniques
- Provide specific service, port, and version information for a given target
- Identify vulnerabilities of accessible services.

Nmap

Nmap is a free open source utility available at: www.insecure.org

- Designed to allow system administrators and curious individuals to scan large networks to determine which hosts are up and what services they are offering
- Can be **DANGEROUS** to IT, SCADA, and PCS systems.

> ***A fast & informative network scanner that can be safely used on isolated nonproduction SCADA/Control System Networks***

Historically, some of the key lessons that have been learned from a testing environment include:

- Various Nmap options have brought down (crashed) different control systems
- Some OSs can't handle multiple incomplete tcp sessions
- Some control systems rely on the services used by some Nmap scans.

What is Nmap?

- Open source tool for network mapping and security auditing.

Why use Nmap?

- Much faster than manual discovery
- Can scan an entire network quickly and offers several options to customize a scan and its results.

While Nmap is commonly used for security audits, many systems and network administrators find it useful for routine tasks such as network inventory, managing service upgrade schedules, and monitoring host or service uptime.

How does Nmap work?

Nmap uses raw IP packets in novel ways to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics.

Nmap is designed and provides options to support a two-stage discovery process: 1) Host discovery and 2) port scanning.

Host Discovery

Host discovery (HD) is a process of identifying active and interesting hosts on a network.

Why does Nmap do HD?

- To significantly reduce the amount of time to complete network scans
- Narrows a set of IP ranges into a list of active or interesting hosts to be port scanned.

How does HD work?

- Uses combination of ARP, ICMP, TCP SYN, TCP ACK packets to identify active hosts.

Nmap default HD settings

- LAN sends ARP scan (**-PR**)
- WAN (Wide Area Network) (privileged) sends TCP ACK packet to Port 80 (**-PA**) and an ICMP echo request query (**-PE**)
- WAN (unprivileged) sends TCP SYN packet (**-PS**) using connect() system call instead of TCP ACK packet.

Nmap common HD options

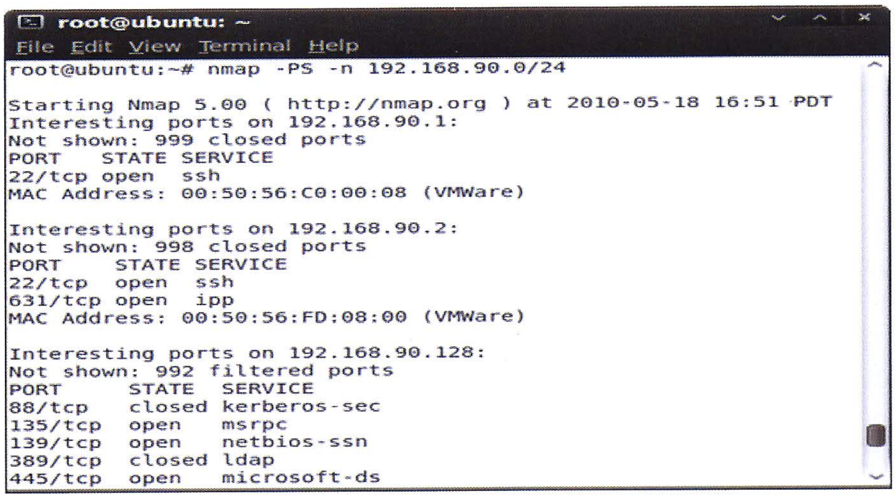
Option	User Level	Speed	Packet Type	Notes
-sn	User	Fast	ICMP echo	Ping only, no port scan
-PA	Root	Fast	TCP Ack	WAN default, port 80, stateless
-PS	User	Fast	TCP Syn	WAN default, port 80,stateful
-PE	Root	Fast	ICMP echo	
-PR	User	Fastest	ARP	LAN default
-PU	Root	Slowest	UDP	Slow, unreliable, firewall
-PN	User	-	-	No ping, no HD

Homeland Security

Note: Zenmap is a GUI interface for Nmap.

Nmap HD example

- Command: **nmap –PS –n 192.168.90.0/24**



```
root@ubuntu:~# nmap -PS -n 192.168.90.0/24

Starting Nmap 5.00 ( http://nmap.org ) at 2010-05-18 16:51 PDT
Interesting ports on 192.168.90.1:
Not shown: 999 closed ports
PORT    STATE SERVICE
22/tcp open  ssh
MAC Address: 00:50:56:C0:00:08 (VMWare)

Interesting ports on 192.168.90.2:
Not shown: 998 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
631/tcp open  ipp
MAC Address: 00:50:56:FD:08:00 (VMWare)

Interesting ports on 192.168.90.128:
Not shown: 992 filtered ports
PORT      STATE  SERVICE
88/tcp    closed kerberos-sec
135/tcp   open   msrpc
139/tcp   open   netbios-ssn
389/tcp   closed ldap
445/tcp   open   microsoft-ds
```

Above is an example of Nmap HD using the command nmap -PS –n 192.168.90.0/24 indicating that no name resolution is to be done and that a TCP SYN packet probe is to be used for the specified hosts.

65,535 ports

Port Scanning

Port scanning is the process of identifying the status of interesting ports on hosts that are discovered on a network.

Why does Nmap do PS?

- To identify ports that are open on a host.

How does PS work?

- Attempts to communicate with each port within a specified set of ports
- Port scans are performed on hosts that were identified as active or interesting during HD.

Nmap port states

While many port scanners have traditionally placed all ports into the open or closed states, Nmap is much more granular. It divides ports into six states:

- Open - Application on target machine is listening for connections or packets on that port
- Closed - No application listening at the moment
- Filtered - Firewall, filter, or other network obstacle is blocking the port so that Nmap cannot tell if the port is open or closed.
- Unfiltered - port is accessible but Nmap not able to determine if open or closed.
- Open | Filtered – unable to determine if open or filtered

For more information on port states see: http://nmap.org/book/man-port-scanning-basics.html

http://www.professormesser.com/nmap/deciphering-nmaps-port-descriptions

- Closed | Filtered – unable to determine if closed or filtered.

Nmap default PS settings

- SYN scan (**-sS**) for privileged users
- Connect scan (**-sT**) for unprivileged users.

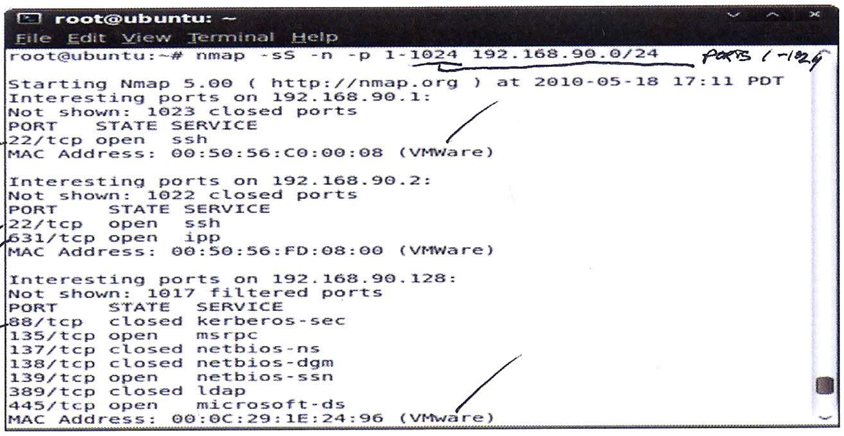
Nmap common PS options

Option	User Level	Packet Type	Notes
-sS	Root	TCP Syn	Privileged default
-sT	User	TCP connect	Uses connect system call
-sA	Root	TCP Ack	Firewall rule sets, stateful?
-sF	Root	TCP Fin	Filter evasion
-sX	Root	TCP FIN, PSH, URG	Filter evasion
-sN	Root	TCP NULL	Filter evasion
-sU	Root	UDP	Find UDP services
-p	-	-	Specify ports to scan

-p - all 65K

The graphic below is the result of using the following command:

- Command: **nmap –sS –n –p 1-1024 192.168.90.0/24**



```
root@ubuntu:~# nmap -sS -n -p 1-1024 192.168.90.0/24

Starting Nmap 5.00 ( http://nmap.org ) at 2010-05-18 17:11 PDT
Interesting ports on 192.168.90.1:
Not shown: 1023 closed ports
PORT    STATE SERVICE
22/tcp open   ssh
MAC Address: 00:50:56:C0:00:08 (VMWare)

Interesting ports on 192.168.90.2:
Not shown: 1022 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
631/tcp open   ipp
MAC Address: 00:50:56:FD:08:00 (VMWare)

Interesting ports on 192.168.90.128:
Not shown: 1017 filtered ports
PORT     STATE  SERVICE
88/tcp   closed kerberos-sec
135/tcp open    msrpc
137/tcp closed netbios-ns
138/tcp closed netbios-dgm
139/tcp open    netbios-ssn
389/tcp closed ldap
445/tcp open    microsoft-ds
MAC Address: 00:0C:29:1E:24:96 (VMWare)
```

Ports 1-1024

The options indicate that a TCP SYN scan is being used against Ports 1 through 1024 for the entire 192.168.90.0 range of addresses. DNS name resolution is not being done.

What are timing and performance options?

- Settings used to control scanning delays, timeouts, retries, and parallelism.

Why use timing and performance options?

- Help speed up scanning process
- Slow down scan to avoid IDS detection

Timing and performance options:

- Manual options are available but templates are usually sufficient
- Template timing options offer throttling abilities not available using manual options.

Nmap timing and performance templates

Option	Nickname	Speed	Notes
-T0	Paranoid	Slowest	IDS avoidance, 5-min packet delay
-T1	Sneaky	Slower	IDS avoidance, 15-sec packet delay
-T2	Polite	Slow	Conserve bandwidth target resources, 0.4 sec packet delay
-T3	Normal	Moderate	Default timing options used by Nmap
-T4	Aggressive	Fast	Maximum dynamic scan delay 10 ms
-T5	Insane	Fastest	Maximum dynamic scan delay 5 ms

Why save your Nmap scan results?

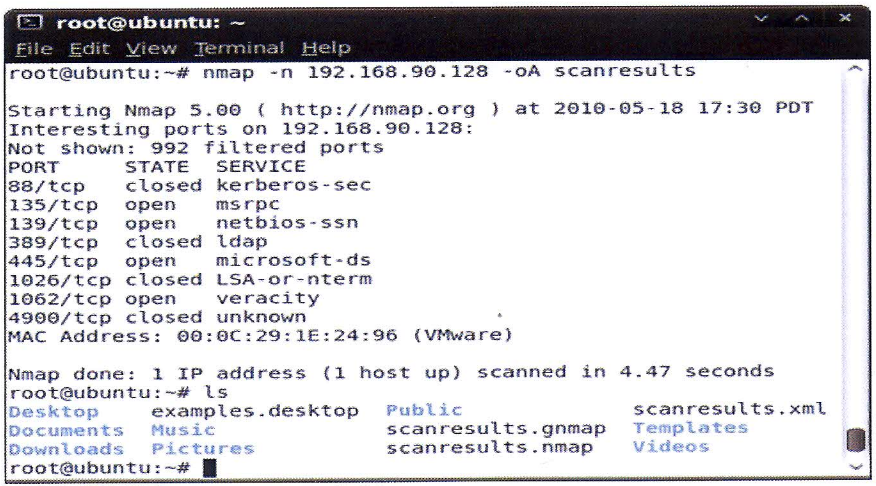
- Easier to analyze and compare scan results (using ndiff)
- Results overflow the console window buffer.

Output options

- **-oN filename.nmap** Output results in normal format
- **-oX filename.xml** Output results in XML format
- **-oG filename.gmap** Output results in grepable format
- **-oA filename.txt** Output results in all formats
- **-v** Verbose output results.

The image below is the results of the following command:

- Command: **nmap –n –oA scanresults 192.168.10.128**



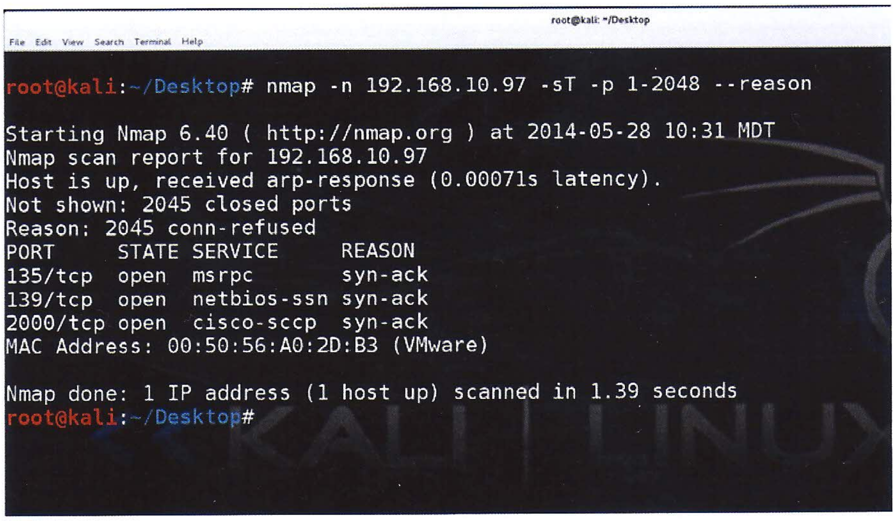
```
root@ubuntu:~# nmap -n 192.168.90.128 -oA scanresults

Starting Nmap 5.00 ( http://nmap.org ) at 2010-05-18 17:30 PDT
Interesting ports on 192.168.90.128:
Not shown: 992 filtered ports
PORT      STATE  SERVICE
88/tcp    closed kerberos-sec
135/tcp   open   msrpc
139/tcp   open   netbios-ssn
389/tcp   closed ldap
445/tcp   open   microsoft-ds
1026/tcp closed LSA-or-nterm
1062/tcp open   veracity
4900/tcp closed unknown
MAC Address: 00:0C:29:1E:24:96 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 4.47 seconds
root@ubuntu:~# ls
Desktop     examples.desktop  Public             scanresults.xml
Documents   Music             scanresults.gnmap  Templates
Downloads   Pictures          scanresults.nmap   Videos
root@ubuntu:~# █
```

If you add the **- -reason** option, Nmap displays the type of packet that determined a port or hosts state and also the reason each host is listed as up or down.

For example, a RST packet from a closed port or an echo replies from a live host. The information Nmap can provide is determined by the type of scan or ping.

The SYN scan and SYN ping (-sS and -PS) (described in the next section) are very detailed, but the TCP connect scan (-sT) is limited by the implementation of the connect system call. The **- -reason** feature is automatically enabled by the debug option (-d) and the results are stored in XML log files even if the **XML output option is NOT selected.** Below is an example of using the **- -reason** option.

- Command: **nmap –n 192.168.10.97 -sT -p 1-2048 --reason**



```
root@kali:~/Desktop# nmap -n 192.168.10.97 -sT -p 1-2048 --reason

Starting Nmap 6.40 ( http://nmap.org ) at 2014-05-28 10:31 MDT
Nmap scan report for 192.168.10.97
Host is up, received arp-response (0.00071s latency).
Not shown: 2045 closed ports
Reason: 2045 conn-refused
PORT     STATE SERVICE     REASON
135/tcp  open  msrpc       syn-ack
139/tcp  open  netbios-ssn syn-ack
2000/tcp open  cisco-sccp  syn-ack
MAC Address: 00:50:56:A0:2D:B3 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.39 seconds
root@kali:~/Desktop#
```

OS & Version Detection

One of Nmap's best-known features is remote OS detection using TCP/IP stack fingerprinting.

What is OS & Version Detection?

- Identifies operating system by looking at packet characteristics
- Identifies the version of a service running on a host.

Why use OS & Version Detection?

- Provides information that could help in the selection of exploits and payloads used against a target.

How does OS Detection work?

- Nmap sends a series of TCP and UDP packets to the remote host and examines every bit in the responses.
- Nmap compares the results to its database of known OS fingerprints and prints out the OS details if there is a match.

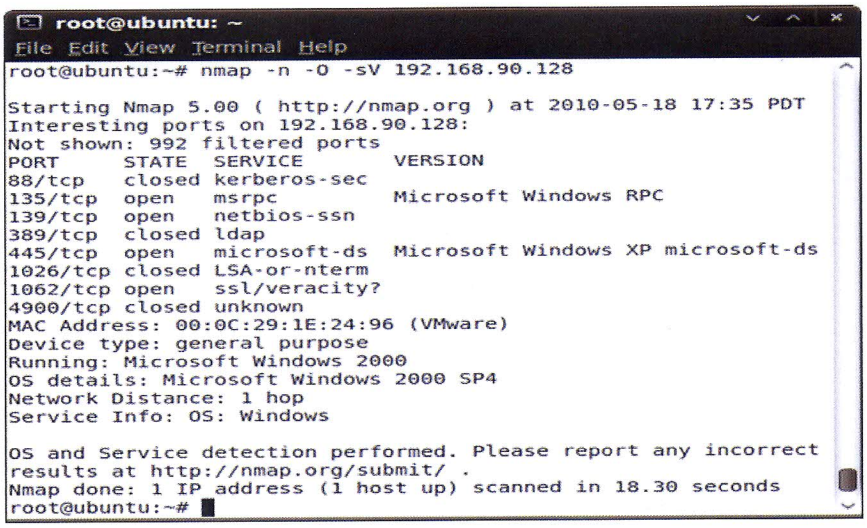
How do Service and Version Detection work?

- After TCP and/or UDP ports are discovered, version detection interrogates those ports.
- Database of probes for querying various services and match expressions to recognize and parse responses.
- Tries to determine application name, version number, hostname, device type, OS family, misc. information.

The image below presents the results of an nmap scan with both OS and version detection enabled.

- Command: **nmap –n –O –sV 192.168.10.128**

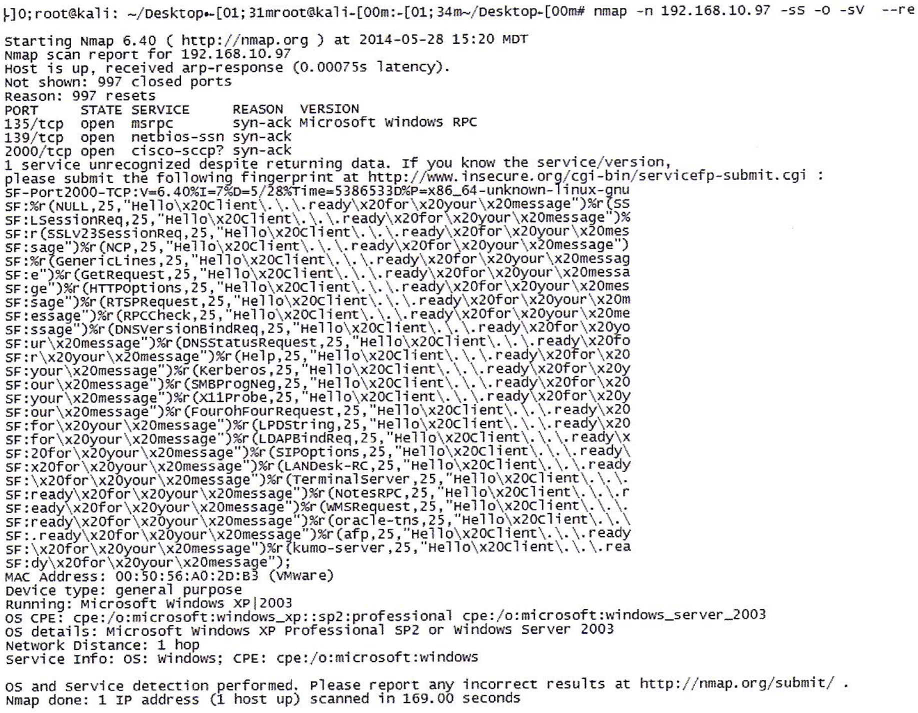
Note that the user is root so the default scan will be either ARP (if on the same network) or TCP ACK.



```
root@ubuntu:~# nmap -n -O -sV 192.168.90.128

Starting Nmap 5.00 ( http://nmap.org ) at 2010-05-18 17:35 PDT
Interesting ports on 192.168.90.128:
Not shown: 992 filtered ports
PORT      STATE  SERVICE        VERSION
88/tcp    closed kerberos-sec
135/tcp   open   msrpc          Microsoft Windows RPC
139/tcp   open   netbios-ssn
389/tcp   closed ldap
445/tcp   open   microsoft-ds   Microsoft Windows XP microsoft-ds
1026/tcp  closed LSA-or-nterm
1062/tcp  open   ssl/veracity?
4900/tcp  closed unknown
MAC Address: 00:0C:29:1E:24:96 (VMware)
Device type: general purpose
Running: Microsoft Windows 2000
OS details: Microsoft Windows 2000 SP4
Network Distance: 1 hop
Service Info: OS: Windows

OS and Service detection performed. Please report any incorrect
results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.30 seconds
root@ubuntu:~# 
```

By adding the --reason option, you obtain additional information about the packets used to determine the Operating System and version information.

Command: **nmap –n 192.168.10.97 -sS -0 -sV --reason**



```
├]0;root@kali: ~/Desktop←[01;31mroot@kali←[00m:←[01;34m~/Desktop←[00m# nmap -n 192.168.10.97 -sS -O -sV  --re

Starting Nmap 6.40 ( http://nmap.org ) at 2014-05-28 15:20 MDT
Nmap scan report for 192.168.10.97
Host is up, received arp-response (0.00075s latency).
Not shown: 997 closed ports
Reason: 997 resets
PORT     STATE SERVICE     REASON  VERSION
135/tcp  open  msrpc       syn-ack Microsoft Windows RPC
139/tcp  open  netbios-ssn syn-ack
2000/tcp open  cisco-sccp? syn-ack
1 service unrecognized despite returning data. If you know the service/version,
please submit the following fingerprint at http://www.insecure.org/cgi-bin/servicefp-submit.cgi :
SF-Port2000-TCP:V=6.40%I=7%D=5/28%Time=5386533D%P=x86_64-unknown-linux-gnu
SF:%r(NULL,25,"Hello\x20Client\.\.\.ready\x20for\x20your\x20message")%r(SS
SF:LSessionReq,25,"Hello\x20Client\.\.\.ready\x20for\x20your\x20message")%
SF:r(SSLv23SessionReq,25,"Hello\x20Client\.\.\.ready\x20for\x20your\x20mes
SF:sage")%r(NCP,25,"Hello\x20Client\.\.\.ready\x20for\x20your\x20message")
SF:%r(GenericLines,25,"Hello\x20Client\.\.\.ready\x20for\x20your\x20messag
SF:e")%r(GetRequest,25,"Hello\x20Client\.\.\.ready\x20for\x20your\x20messa
SF:ge")%r(HTTPOptions,25,"Hello\x20Client\.\.\.ready\x20for\x20your\x20mes
SF:sage")%r(RTSPRequest,25,"Hello\x20Client\.\.\.ready\x20for\x20your\x20m
SF:essage")%r(RPCCheck,25,"Hello\x20Client\.\.\.ready\x20for\x20your\x20me
SF:ssage")%r(DNSVersionBindReq,25,"Hello\x20Client\.\.\.ready\x20for\x20yo
SF:ur\x20message")%r(DNSStatusRequest,25,"Hello\x20Client\.\.\.ready\x20fo
SF:r\x20your\x20message")%r(Help,25,"Hello\x20Client\.\.\.ready\x20for\x20
SF:your\x20message")%r(Kerberos,25,"Hello\x20Client\.\.\.ready\x20for\x20y
SF:our\x20message")%r(SMBProgNeg,25,"Hello\x20Client\.\.\.ready\x20for\x20
SF:your\x20message")%r(X11Probe,25,"Hello\x20Client\.\.\.ready\x20for\x20y
SF:our\x20message")%r(FourOhFourRequest,25,"Hello\x20Client\.\.\.ready\x20
SF:for\x20your\x20message")%r(LPDString,25,"Hello\x20Client\.\.\.ready\x20
SF:for\x20your\x20message")%r(LDAPBindReq,25,"Hello\x20Client\.\.\.ready\x
SF:20for\x20your\x20message")%r(SIPOptions,25,"Hello\x20Client\.\.\.ready\
SF:x20for\x20your\x20message")%r(LANDesk-RC,25,"Hello\x20Client\.\.\.ready
SF:\x20for\x20your\x20message")%r(TerminalServer,25,"Hello\x20Client\.\.\.
SF:ready\x20for\x20your\x20message")%r(NotesRPC,25,"Hello\x20Client\.\.\.r
SF:eady\x20for\x20your\x20message")%r(WMSRequest,25,"Hello\x20Client\.\.\.
SF:ready\x20for\x20your\x20message")%r(oracle-tns,25,"Hello\x20Client\.\.\
SF:.ready\x20for\x20your\x20message")%r(afp,25,"Hello\x20Client\.\.\.ready
SF:\x20for\x20your\x20message")%r(kumo-server,25,"Hello\x20Client\.\.\.rea
SF:dy\x20for\x20your\x20message");
MAC Address: 00:50:56:A0:2D:B3 (VMware)
Device type: general purpose
Running: Microsoft Windows XP|2003
OS CPE: cpe:/o:microsoft:windows_xp::sp2:professional cpe:/o:microsoft:windows_server_2003
OS details: Microsoft Windows XP Professional SP2 or Windows Server 2003
Network Distance: 1 hop
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 169.00 seconds
```

Nmap address schemes

Targets can be specified in ranges or by using a netmask called the Classless Inter-domain Routing (CIDR) notation.

- 1.2.3.1-254

    - All 254 possible IP addresses on this subnet
- 1.2.3.0/24
    - Equivalent to the above but signifying a Class C address block
- 1.2.1-4.1-254
    - Ranges are allowed for subnets as well
- 1.2.0.0/16
    - The 16-bit netmask will scan the entire Class B address block.

The target IP's and/or networks can also be read from an ASCII file. Simply generate the list of hosts to scan and pass that filename to Nmap as an argument to the **-iL** *filename* option.

The entries can be in any of the formats accepted by Nmap on the command line (IP address, hostname, CIDR, IPv6, or octet ranges).

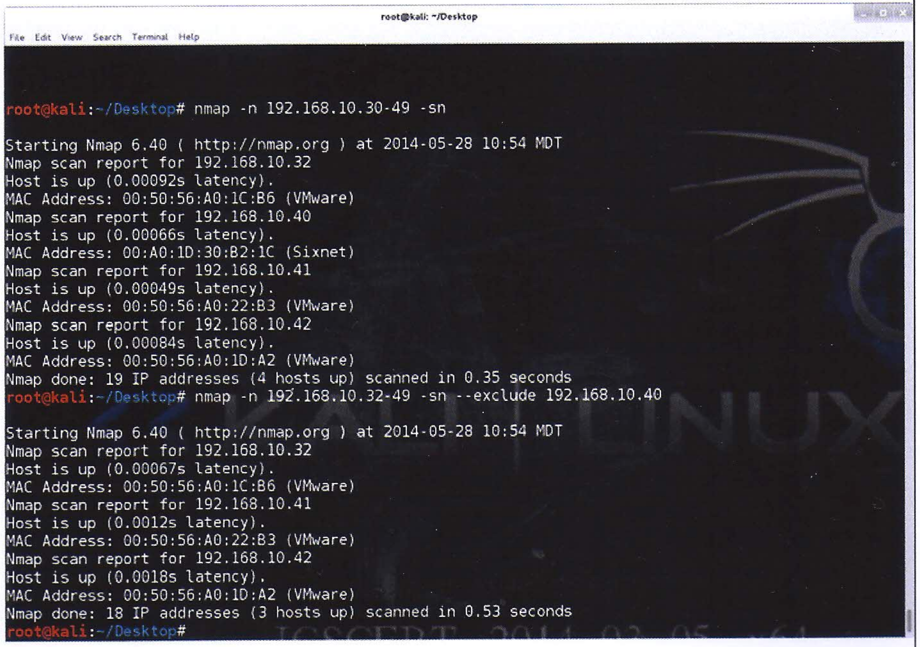
Each entry must be separated by one or more spaces, tabs, or newlines.

You can specify a hyphen (-) as the filename if you want Nmap to read hosts from standard input rather than an actual file. This option is handy if the list is being generated by another utility. The input file may contain comments that start with # and extend to the end of the line. This option also allows you to skip host you do not want to scan.

You can also explicitly exclude hosts by using the --exclude host1[,host2[,…]] option on the command line. If you prefer to use a file, you can create it using the same format as the input list described above and use the **- -excludefile** *exclude_filename option.*

NOTE: This is a hyphenhyphenexclude in both cases, not a long dash!!

Command: **nmap –n 192.168.10.32-49 -sn --exclude 192.168.10.40**



```
root@kali:~/Desktop# nmap -n 192.168.10.30-49 -sn

Starting Nmap 6.40 ( http://nmap.org ) at 2014-05-28 10:54 MDT
Nmap scan report for 192.168.10.32
Host is up (0.00092s latency).
MAC Address: 00:50:56:A0:1C:B6 (VMware)
Nmap scan report for 192.168.10.40
Host is up (0.00066s latency).
MAC Address: 00:A0:1D:30:B2:1C (Sixnet)
Nmap scan report for 192.168.10.41
Host is up (0.00049s latency).
MAC Address: 00:50:56:A0:22:B3 (VMware)
Nmap scan report for 192.168.10.42
Host is up (0.00084s latency).
MAC Address: 00:50:56:A0:1D:A2 (VMware)
Nmap done: 19 IP addresses (4 hosts up) scanned in 0.35 seconds
root@kali:~/Desktop# nmap -n 192.168.10.32-49 -sn --exclude 192.168.10.40

Starting Nmap 6.40 ( http://nmap.org ) at 2014-05-28 10:54 MDT
Nmap scan report for 192.168.10.32
Host is up (0.00067s latency).
MAC Address: 00:50:56:A0:1C:B6 (VMware)
Nmap scan report for 192.168.10.41
Host is up (0.0012s latency).
MAC Address: 00:50:56:A0:22:B3 (VMware)
Nmap scan report for 192.168.10.42
Host is up (0.0018s latency).
MAC Address: 00:50:56:A0:1D:A2 (VMware)
Nmap done: 18 IP addresses (3 hosts up) scanned in 0.53 seconds
root@kali:~/Desktop#
```

P0f – Passive Traffic Fingerprinting/OS & Version Detection

P0f is a tool that utilizes an array of sophisticated, purely passive traffic fingerprinting mechanisms to identify the players behind any incidental TCP/IP communications (often as little as a single normal SYN) without interfering in any way. The process is completely passive and does not generate any suspicious network traffic.

Version 3 is a complete rewrite of the original codebase, incorporating a significant number of improvements to network-level fingerprinting, and introducing the ability to reason about application-level payloads (e.g., HTTP).

Some of p0f's capabilities include:

- Highly scalable and extremely fast identification of the operating system and software on both endpoints of a vanilla TCP connection - especially in settings where NMap probes are blocked, too slow, unreliable, or would simply set off alarms.
- Measurement of system uptime and network hookup, distance (including topology behind NAT or packet filters), user language preferences, and so on.
- Automated detection of connection sharing / NAT, load balancing, and application-level proxying setups.
- Detection of clients and servers that forge declarative statements such as *X-Mailer* or *User-Agent*.

The tool can be operated in the foreground or as a daemon, and offers a simple real-time API for third-party components that wish to obtain additional information about the actors they are talking to.

Common uses for p0f include reconnaissance during penetration tests; routine network monitoring; detection of unauthorized network interconnects in corporate environments; providing signals for abuse-prevention tools; and miscellaneous forensics.

Since this tool listens passively to network traffic to work, it is safe to use in environments where active scanning or discovery is not feasible

Below is an example of a p0f scan.

```
p0f -f /etc/p0f/p0f.fp -i eth1 -p -o p0f.out

Output written to screen:

.-[ 1.2.3.20/3701 -> 1.10.11.179/1337 (syn) ]-
|
| client    = 1.2.3.20/3701
| os        = Windows NT kernel
| dist     = 0
| params   = generic
| raw_sig  = 4:128+0:0:1460:mss*44,0:mss,nop,nop,sok:df,id+:0
|
`----

.-[ 204.138.254.145/56301 -> 1.1.1.20/80 (http request) ]-
|
| client   = 204.138.254.145/56301
| app      = wget
| lang     = none
| params   = none
| raw_sig  = 0:User-Agent,Accept=[*/*],Host,Connection=[Keep-Alive]:Accept-
Encoding,Accept-Language,Accept-Charset,
             Keep-Alive:Wget/1.11.4 Red Hat modified
|
`----

.-[ 204.138.254.143/53395 -> 1.2.3.6/80 (http response) ]-
|
| server   = 1.2.3.6/80
| app      = Apache 2.x
| lang     = none
| params   = none
| raw_sig  = 1:Date,Server,X-Powered-By=[PHP/5.4.6-1ubuntu1.1],?Set-
Cookie,?Expires,?Cache-Control,?Pragma,?Vary,
             ?Content-Length,Keep-Alive=[timeout=5, max=100],Connection=[Keep-
Alive],Content-Type:Accept-Ranges:
             Apache/2.2.22 (Ubuntu)
|
`----
```

Output written to file (-o p0f.out)

```
[2015/02/25 16:59:25]
mod=syn|cli=1.2.3.20/3701|srv=1.10.11.179/1337|subj=cli|os=Windows NT
kernel|dist=0|params=generic|
raw_sig=4:128+0:0:1460:mss*44,0:mss,nop,nop,sok:df,id+:0

[2015/02/25 16:59:26] mod=http request|cli=204.138.254.145/56301|
srv=1.1.1.20/80|subj=cli|app=wget|lang=none|params=none|
raw_sig=0:User-Agent,Accept=[*/*],Host,Connection=[Keep-Alive]: Accept-
Encoding,Accept-Language,Accept-Charset,Keep-Alive:
Wget/1.11.4 Red Hat modified

[2015/02/25 16:59:26] mod=http
response|cli=204.138.254.145/56301|srv=1.1.1.20/80|subj=srv|app=???|lang=none|
params=none|
raw_sig=1:Server,Connection=[keep-alive],Date,Content-Type,Accept-
Ranges=[bytes],?Last-Modified,?ETag,?Content-Length
Keep-Alive:Microsoft-IIS/5.0
```

See http://nmap.org/book/man.html or http://linux.die.net/man/1/nmap for online nmap manual pages.

ICS challenges

- Scans can cause computer systems to restart.
- Scans can cause embedded devices to freeze or lose configuration and in some severe cases requires vendor involvement.
- Nmap considerations:
  - Use connect scan (-sT) to prevent dangling connections.
  - Don't use OS (-O) and Version Detection (-sV).
  - Slow the scan down by reducing the rate at which packets are being generated and sent by Nmap. For example, specify the option **-T2**or slower on the Nmap command line.
  - Consider using exclusion lists (--exclude or --excludefile) to keep Nmap from scanning risky hosts (ones that could be damaged by a scan).

Active Discovery: Nmap Hands-on Exercise

Nmap

Feel free to launch Wireshark or tcpdump during the nmap exercises in order to analyze the network traffic generated. Type the following nmap command in a command shell to do Host Discovery of the network using a TCP SYN Ping. The first command will change the current directory to the desktop.

```
cd ~/Desktop
nmap  -PS  -n   <your network>.1-99
```

The result of this command is a list of the hosts that responded during the scan. A report is also in the output of the status of ports that were checked. Nmap scans the most common 1,000 ports for each protocol. Next, add the "-sn" option to the previous command.

Note: Refer to the Linux nmap man page (type "man nmap" on the command line) for details on nmap options.

Note: You can use the 'up' arrow to recall the previous command and the 'left' arrow to insert text.

```
nmap -PS -n -sn <your network>.1-99
```

Again, the result is a list of hosts that responded to the scan. However, there are no port status entries in this output result. The -sn option tells Nmap not to do a port scan after host discovery, and only print out the available hosts that responded to the scan. System administrators often find this option valuable. It can easily be used to count available machines on a network or monitor server availability. This is often called a ping sweep and is more reliable than pinging the broadcast address because many hosts do not reply to broadcast queries.

Using the same command as before, add the Nmap option "--reason" to the end of the command.

```
nmap -PS -n -sn <your network>.1-99 --reason
```

This time you see that for every host that was determined to be up, there is additional text in the "Host is up" line that reads something like "received arp-response." The "- -reason" option tells Nmap to report the reason each port is set to a specific state and the reason each host is up or down by displaying the type of packet that determined the port or host state. When a privileged user tries to scan targets on a local ethernet network, ARP requests are used unless "**- -send-ip"** was specified. So, let's use the send-ip switch by typing the following command and see what difference the send-ip switch makes.

```
nmap -PS -n -sn <your network>.1-99 --reason  --send-ip
```

This time the "Host is up" line indicated that TCP packets such as "syn-ack" or "reset" where received to determine the host status. This is because the send-ip switch forced the packet type specified by the -PS switch to be used rather than using the default ARP packet (-PR) on the local area network (LAN).

Try some of the other Host Discovery options available in Nmap while analyzing the traffic with Wireshark. Look for differences in the packets that are being used to perform the different Host Discovery scans.

The next exercise is to run a Port Scan using a TCP SYN scan with the following nmap command:

```
nmap -n -sS <your network>.1-99
```

The result of this command should be a list of hosts and a status of the common 1,000 ports for the TCP protocol for that host. To get a status of all possible ports add the –p– option to the command above. The –p option specifies which ports to scan and the – after the p is shorthand for 1-65,535. Be warned that whenever all 65,535 ports are scanned, it takes significantly longer to complete the scan.

The next exercise demonstrates the difference between two of the Timing and Performance options. First run a ping sweep scan using –T2 with the following command:

```
nmap -T2 -sn -n  <your network>.1-99
```

Now run the same command with a –T5.

```
nmap -T5 -sn -n <your network>.1-99
```

There should have been a significant difference in the amount of time it took to run these scans. The first scan sends packets at a rate of one packet every 0.400 seconds compared to one packet every 0.005 seconds (5 ms) in the second scan.

Nmap provides a variety of ways to save scan results including normal ASCII, XML, Grepable, and the ability to save in all formats with a single option. Type the following command to perform a default scan and save the results in all possible formats.

```
nmap -n -oA filename <your network>.1-99
```

A listing of the current directory should show three files with the following extensions .gnmap, .nmap, and .xml.

The final exercise demonstrates the ability of nmap to do Operating System (OS) detection and Service Version Detection.

Type the following command, where –O is OS detection and –sV is Version Detection.

```
nmap -n -O -sV <your network>.1-99
```

The results of this scan should provide you information regarding the OS of the host and Service Versions for any ports that were open. The following two commands are more examples for you to try that put everything together into a single command. Feel free to come up with some of your own scans using the options that have been discussed. These commands will take some time so you might want to consider running the command against a host (or hosts) of interest rather than the entire range (1-99) of hosts.

For example:

nmap –n –sS <your network> .21<your network>.36

```
nmap -n  -sS -O -sV -T4 <your network>.1-99
nmap -n -sS -O -sV -p- -T4 <your network>.1-99
```

Remember to use the man pages for nmap (“man nmap”) if you are not sure what options are available or if you want more information regarding a particular option or refer to the referenced online documentation.

Post-exercise analysis

- How many hosts did you discover?

- What interesting ports did you find open?

- How does timing and performance affect your scan times?

- How do the output results differ?

- What OS and service versions did you find?

Nmap review

- Nmap is a network discovery tool that can be used for identifying the systems *currently* connected to your network
- Nmap allows you to audit what services are running on the identified hosts
- Nmap can be **DANGEROUS** to ICSs and embedded devices.

OpenVAS - Open Vulnerability Assessment System

- Open source fork of Nessus
- Can be **DANGEROUS** to ICS systems
- Plug-in modules for various ICS protocols
- Security auditing tool consists of two parts

Server

The Server, openvassd, is in charge of the attacks.

Client

The OpenVAS-Client provides an interface to the user.

The "Nessus" Project was started by Renaud Deraison in 1998 to provide to the Internet community a free remote security scanner. On October 5, 2005, Tenable Network Security, the company Renaud Deraison co-founded, changed Nessus 3 to a proprietary (closed source) license.

Nessus ICS plugins (not on CD)

Below are some of the ICS plugins that are available for Nessus. These plugins are not on your Kali CD.

- Areva/Alstom Energy Management System
- DNP3 Binary Inputs Access
- DNP3:
  - Link Layer Addressing DNP3
  - Unsolicited Messaging
- ICCP
  - ICCP/COTP Protocol
  - ICCP/COTP
  - TSAP Addressing
  - LiveData ICCP Server
- Matrikon OPC Explorer
- Matrikon OPC Server for ControlLogix

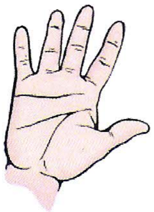
- Matrikon OPC Server for Modbus
- Modbus/TCP:
  - Coil Access
  - Discrete Input Access Programming
  - Function Code Access
- Modicon:
  - Modicon PLC CPU Type
  - PLC Default FTP Password
  - PLC Embedded HTTP Server
  - PLC HTTP Server Default Username/Password
  - PLC Telnet Server
  - IO Scan Status
- Modbus Slave ModeModicon PLC Web Password Status
- National Instruments Lookout
- OPC DA Server/OPC Detection/OPC HDA Server
- Siemens S7-SCL
- Siemens SIMATIC PDM
- Siemens-Telegyr ICCP Gateway
- Sisco OSI/ICCP Stack
- Sisco OSI Stack Malformed Packet Vulnerability
- Tamarack IEC 61850 Server

OpenVAS plugin options

- Backdoors
- CISCO
- Database
- FTP
- Gain a shell remotely
- Gain root remotely
- General
- Misc
- RPC (Remote Procedure Call)
- Remote file access
- Settings
- Web Server
- Windows
- Windows: MS Bulletins

  Windows: User management.

Active Discovery: Nessus and OpenVAS Hands-on Exercise



During this activity, you will:

- Review "Scan Reports" located in the directory /root/Desktop/Scan_Reports for output from Nessus and OpenVAS
- Identify specific vulnerabilities for network hosts
- Complete your network map.

Post-exercise review

- What information does OpenVAS provide that you didn't find with Nmap?
- What different types of security problems did you discover?
- Were there any false-positives and how did you identify them?

Review of Session 3

- Understanding of basic networking concepts necessary for doing active and passive network discovery
- Manual and automated techniques for discovery
- Understand the potential impacts from automated tools like Nessus, Nmap, etc.